**Practica 4:**

**Construcción de árboles de sintaxis abstracta**

**Grupo 11:**

**Youssef El Faqir El Rhazoui**

**Enrique Ávila Rodríguez**

1. **Conjunto de funciones constructoras**

**Prog:** Sec\_Dec **x** Sec\_Ins 🡪 Prog

**Sec\_Dec:** LDs 🡪 Prog

**Sec\_Ins:** LIs 🡪 Prog

**LD\_simp:** String **x** String 🡪 LDs

**LD\_comp:** String **x** String **x** LDs 🡪 LDs

**LI\_simp:** String **x** Exp 🡪 LIs

**LI\_comp:** String **x** Exp **x** LIs 🡪 LIs

**Mas:** Exp **x** Exp 🡪 Exp

**Menos:** Exp **x** Exp 🡪 Exp

**And:** Exp **x** Exp 🡪 Exp

**Or:** Exp **x** Exp 🡪 Exp

**Distinto:** Exp **x** Exp 🡪 Exp

**Igual:** Exp **x** Exp 🡪 Exp

**Menor\_que:** Exp **x** Exp 🡪 Exp

**Menor\_igual\_que:** Exp **x** Exp 🡪 Exp

**Mayor\_que:** Exp **x** Exp 🡪 Exp

**Mayor\_igual\_que:** Exp **x** Exp 🡪 Exp

**Por:** Exp **x** Exp 🡪 Exp

**Div:** Exp **x** Exp 🡪 Exp

**Not:** Exp 🡪 Exp

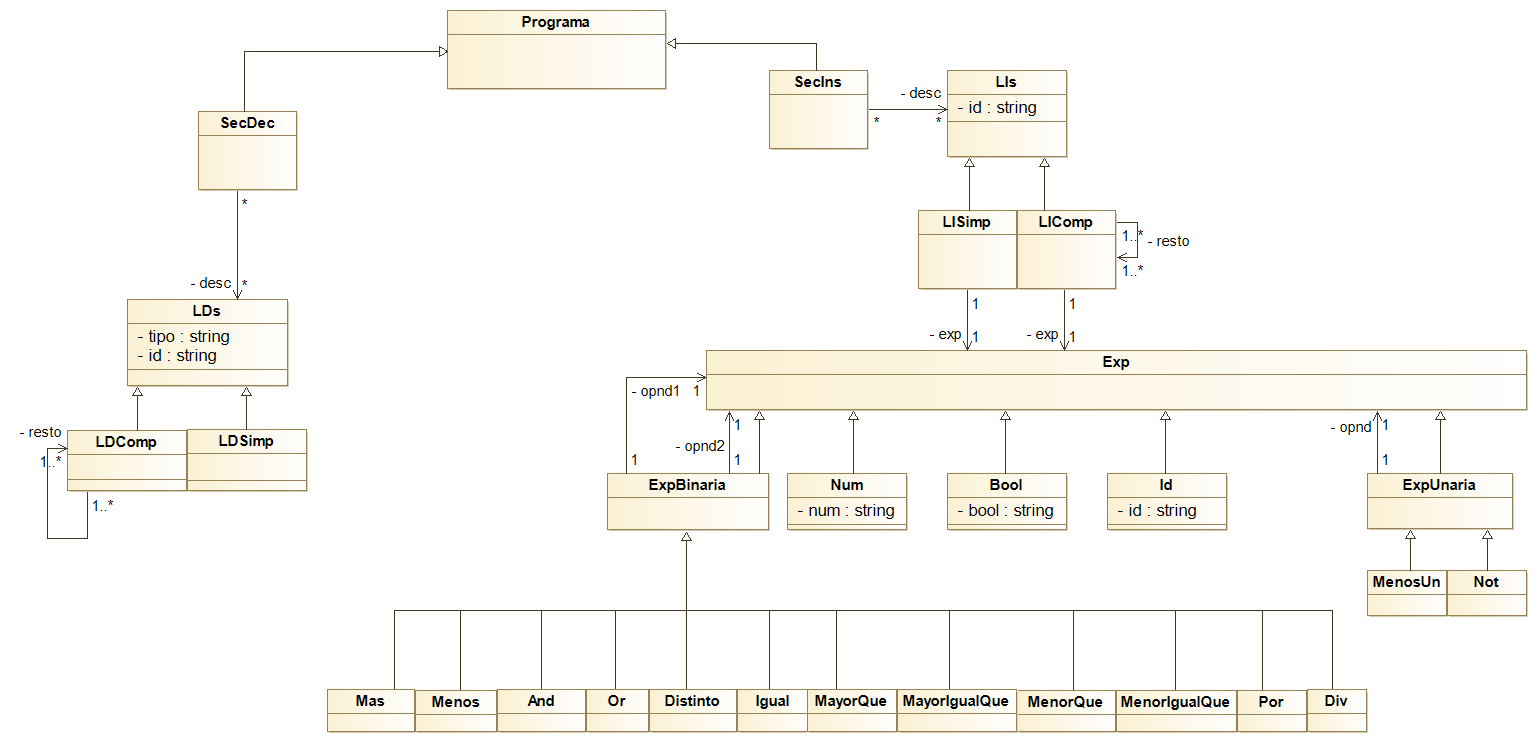
**Menos\_unario:** Exp 🡪 Exp

**Num:** String 🡪 Exp

**Bool:** String 🡪 Exp

**Id:** String 🡪 Exp

1. **Diagrama de clases**



1. **Gramática de atributos**

Prog 🡪 Sec\_Dec Sec\_Ins

Sec\_Dec 🡪 LDs; D

Sec\_Dec.a = ldCompuesta(LDs.a, D.tipo, D.id)

Sec\_Dec 🡪 D

Sec\_Dec.a = ldSimple(D.tipo, D.id)

Sec\_Ins 🡪 LIs; I

Sec\_Ins.a = liCompuesta(LIs.a, I.id, I.exp)

Sec\_Ins 🡪 I

Sec\_Ins.a = liSimple(I.id, I.exp)

D 🡪